Xilinx Rapid Development Embedded Design

Embedded Software 4

EMBD-88080 (v1.0)

Course Specification

Course Description

This course provides Xilinx MPSoC and ACAP firmware and application developers powerful tools and techniques to hit the ground running on your Xilinx embedded design projects.

Using custom labs developed by Morgan Advanced Programmable Systems, Inc., engineering participants will be challenged to learn little known but simple techniques to significantly increase the number of design iterations per day, while covering everything from custom IP, interrupts, baremetal Embedded Linux devices drivers models, Linux applications development using various device driver models.

Furthermore, you will be reminded or learn interrupts, interrupt latency, latency measurement, cache, tightly coupled memory, linker scripts, DDR memory reservation, concurrent development and debugging of Linux applications, Baremetal Applications, and hardware using Vivado Logic Analyzer and Virtiual Input Output. This will ultimately allow you to remotely debug an MPSoC or RFSoC system whether it is down the hall, or across the world connected by Virtual Private Network.

These custom labs are based on the real-world situations developers in industry must deal with, such as revision control, rapid verification (HW and SW), remote development, concurrent development, troubleshooting, and adapting to evolving requirements.

Level – Embedded Software 4

Course Details

- 4 days live instructor led training (in person or online)
- 17 lectures
- 9 labs

Price – $3,200 or 32 Xilinx Training Credits

Course Duration – 4 days

Course Part Number – EMBD-88080

Who Should Attend? – Embedded systems developers interested in customizing in rapid prototyping and development in Xilinx SoCs

Prerequisites

- Familiarity with Linux or Unix and an ability to type accurately. This training has highly technical labs.
- C and Embedded baremetal design using any processor.
- Familiarity with device drivers
- Familiarity with computer architecture concepts (virtual memory, cache, DMA, etc.)
- Familiarity with batch files or shell scripts (Tcl, Bash).
- Coding in VHDL is not required to complete this course, as the point of the course is to guide you through various paths, and selecting the most productive paths for embedded design.
- Video: Linux Tutorial – Basic Command Line
- Video: Vitis Introduction for Embedded Software Development
- Video: Xilinx Embedded Linux Build Flows: Petalinux Tools
- Wikipedia: Make and BitBake

Software Tools

- PetaLinux Tools 2020.1
- Vivado® Design Suite 2020.1
- Vitis™ unified software platform 2020.1

Hardware

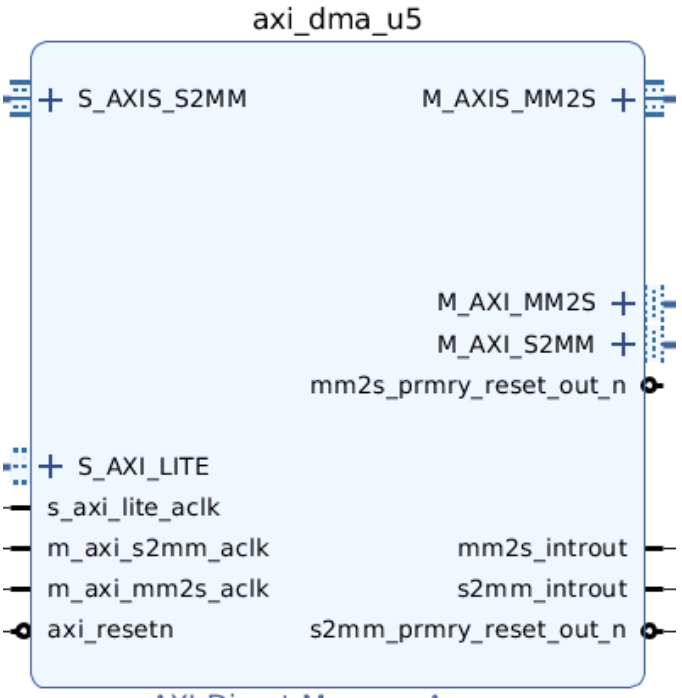
- Architecture: Zynq® UltraScale+™ MPSoC
- Demo board: Zynq UltraScale+ MPSoC ZCU104 board (provided during training, or use your own).

* This course focuses on the Zynq UltraScale+ MPSoC. Contact Morgan Advanced Programmable Systems, Inc. for the specifics of the in-class lab board or other customizations.

This course provides embedded systems developers experience with creating Custom Hardware, baremetal/R5 and embedded Linux/A53 system targeting Xilinx MPSoCs and RFSoC using the PetaLinux tools. The techniques learned here can also be easily applied to lower cost Zynq-7000 embedded designs (both baremetal and Embedded Linux) as well as the Xilinx MicroBlaze processors in any FPGA.

Using custom and standard IP from the Vivado IP catalog, this course provides experience with writing custom and using standard drivers for devices in the PL (Programmable Logic), such as:

oneSecTimer_v1.0 (Pre-Production)



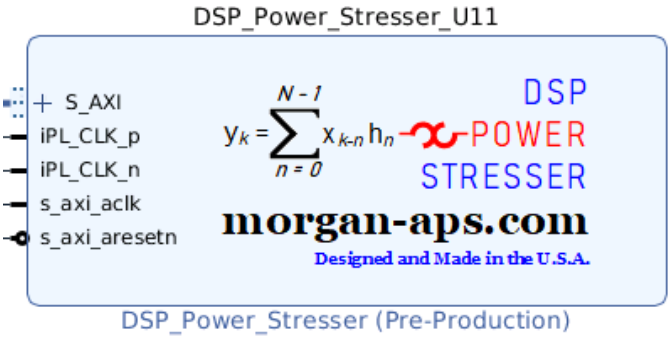
AXI Direct Memory Access

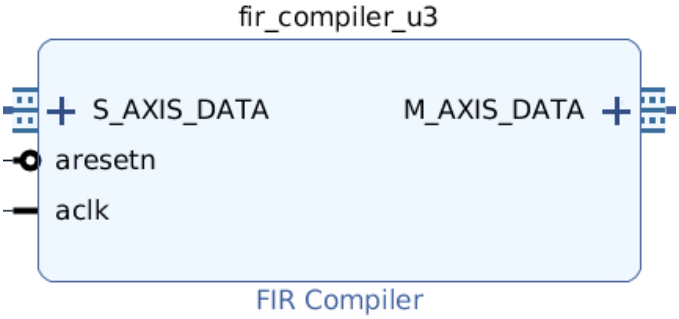
Xilinx Rapid Development Embedded Design

Embedded Software 4

EMBD-88080 (v1.0)

Course Specification





The course provides experience with:

- Cursory custom Hardware development with upfront tricks to help development/debug cycles (no hardware design experience necessary).
- Creating and packaging IP in Vivado
- Baremetal driver and application development in Vitis, polling and interrupt driven.
- Creating and debugging Linux drivers for custom hardware in Vitis SDK.
- Using PetaLinux/Yacto tools to build an embedded Linux OS
- Creating Embedded Linux template applications in PetaLinux
- Various device driver models with labs
- Developing and troubleshooting applications in Vitis SDK.
- Building the environment and booting the system using the Arm® processors available in Xilinx SoCs
- Customizing the root file system for rapid development and safe management of multiple card updates.
- Configuring and exploiting network utilities for rapid development.
- Trouble shooting hardware and software applications simultaneously using Vivado Logic Analyzer and the Vitis SDK.

The primary focus is on embedded Linux/A53 driver and application development as well as baremetal R5 driver and application in conjunction with the Xilinx tool flow. However, the techniques employed can be used with MicroBlaze and A72 processors as well.

After completing this comprehensive training, you will have the necessary skills to:

- Build a custom piece of Intellectual Property (IP) in the programmable logic with AXI and interrupt interfaces.
- Write a baremetal driver and application to use that custom IP using polling and interrupt driven modes in Vitis SDK.
- Build a PetaLinux project to aid in field updates
- Build a PetaLInux project for driver and application development.
- Control the custom hardware using polling and interrupt driven modes of Embedded Linux applications.
- Control the custom hardware using polling and interrupt driven modes of baremetal applications.
- Characterize latency of various interrupt driven modes (measure interrupt latencies: minimum, maximum, mean, and variance) in order to architect your system based on system requirements for both baremetal and Linux drivers/applications.
- Develop and deploy a UIO Framework driver and application (lab)
- Develop and deploy a Kernel module driver for the custom IP. (lab)
- Develop and deploy a shared object/dynamically linked library. (lab)
- Develop and deploy a statically linked library creation and development (lab)
- Control hardware through various mechanisms (UIO Framework, /dev/mem, custom drivers and ioctl). (lab)
- Learn what features to architect into custom devCards for rapid development and field updates.
- Increase productivity over standard training from 4 iterations per day to dozens of iterations per day, shaving months off a typical project. (lab)
- Learn practical techniques and tricks to greatly reduce the build times for Vivado and PetaLinux. (lab/lecture)
- Learn how to automate while reducing risks of typographical errors. The scripts developed in this training make the cost of the entire course worth the investment in time. (lab/lectue)
- Learn how to configure PeltaLinux for board/PCB level customizations not captured in Vivado IPI and driver customizations.
- Explain the concept of an embedded Linux kernel
- Describe various Linux device driver options and their tradeoffs
- Create a PetaLinux project to configure and build an image (lab)
- Create a working Arm processor-based Linux system using the Vivado Design Suite and PetaLinux tools (lab)
- List various hardware interfacing options available for the Arm processor
- Describe the Linux device driver architecture.

Course Outline

The following agenda and sequencing of this course is still in development, and subject to change. More material will be provided than can be completed in four days. Some of the material will be for self-study and reference.

Day 1

- **Introduction to MPSoC**

  Introduces the Zynq US+ MPSoC, including a brief architectural overview. We will build a two Vivado Projects, one of which will

have custom interrupt driven Hardware {Lectures, custom real world labs}

- **Developing an interrupt driven application to control custom hardware using Vitis and R5 processors.**

  Introduces Vitis for bare-metal driver and application development. Introduces rapid development using TCF-Framework and remote debugging (HW and SW). {Lecture, custom real-world lab}

Day 2

- **Introduction to Embedded Linux and Petalinux Components**

  Introduces embedded Linux, including a brief architectural overview, as well as some of the reasons for its rising popularity as an embedded OS. Also introduces the concept of toolchains and cross-compilation. Describes the various components required for embedded Linux platforms (including the kernel image, root file system, and boot loaders) and how the components affect the booting of Linux on these platforms. {Lecture, recipe lab}

- **Driving the PetaLinux Tool**

  Covers the functionality, inputs, and outputs of the PetaLinux tools as well as the project directory structure generated by the PetaLinux tools. Basic PetaLinux commands are also introduced. {Lecture, recipe lab}

Days 3 and 4

- **Building Embedded Linux OSes for rapid development**

  Learn to create and customize embedded Linux builds for rapid Linux application development and safe field updates. Learn to control custom hardware with Linux applications using polling, interrupts.

  Debugging a System Project in Vitis with Linux applications, bare meta applications while simultaneously debugging custom hardware using Vivado Logic Analyzer and Virtual Input/Output.

  Troubleshooting hardware via Vivado Logic Analyzer (VLA) and VIO (Virtual Input/Output) while debugging Linux applications in Vitis, empowering you to discover design defects in hours instead of weeks. {Custom real-world labs}

Self-Study Day 1:

- **PetaLinux Tool Design Flow**

  Provides a brief description of the PetaLinux tool design flow and describes in detail various PetaLinux commands (including PetaLinux-create, PetaLinux-config, PetaLinux-build, PetaLinux-package, and PetaLinux-boot) and their example use cases. {Lecture}

- **PetaLinux Application Development**

  Introduces core concepts for developing, customizing, and running software applications in an embedded Linux environment. {Lecture, Lab}

- **Customizing the Project**

  Analyzes different configuration options provided by the PetaLinux tool for firmware version, rootfs type, boot image storage, and primary flash partition. Also describes external file system boot configuration. {Lecture}

- **Customizing the Root File System**

  Provides a brief description on customizing the rootfs for embedded Linux components such as libraries, applications, modules, layers, recipes, and packages. {Lecture}

- **Networking and TCP/IP**

  Discusses how the TCP/IP networking stack can be used to improve productivity during embedded product development by supporting network data communication, network control/status management, and firmware and hardware upgrades. {Lecture, Lab}

- **PetaLinux Booting and Packaging**

  Describes how to package and then boot a PetaLinux image via QEMU, SD card, JTAG, and TFTP. {Lecture}

Self-Study Day 2

- **PetaLinux Application Debugging**

  Describes how to debug software applications running on an Arm processor using the system debugger (TCF agent) or GNU debugger (GDB). {Lecture, Lab}

- **Upgrading the Workspace**

  Describes the petalinux-upgrade command and how to upgrade PetaLinux project software components without changing the host tool components. {Lecture}

- **Basic Hardware Design Process with the Vivado Design Suite**

  Describes the complete board bring-up process, which includes the hardware design as well as Linux image creation for the hardware. {Lecture, Lab}

- **Linux Device Drivers Overview**

  Provides a brief overview on Linux device drivers and their requirements. Also describes what a device tree is and how it is generated. {Lecture}

- **User Space I/O and Loadable Kernel Modules**

  Introduces two lightweight approaches for accessing the physical memory of devices from user space: direct access through the `dev/mem` virtual device and the user space I/O framework. Also covers the role and usage loadable kernel modules. {Lecture, Lab}

- **Custom Hardware Development (Self-study)**

  Describes the Create and Package IP Wizard and how it can be used to create a variety of architectural options for interfacing a system with custom processing hardware. {Lecture, Lab}

- **Custom Driver Development (Self-study)**

  Discusses device driver options to match custom hardware devices and how to use the provided interfaces to read and write to the devices. {Lecture, Lab}

- **PetaLinux: Advanced Configurations**

  Reviews how modify advanced configuration settings using the PetaLinux tool. These configurations include including selecting the Linux components for the build, enabling automatic configuration for a selected component, customizing how the Linux system interacts with the underlying hardware platform. {Lecture}

Custom Lab overview

The lab is composed of the following major points:

- Non-trivial custom hardware development with interrupts. No RTL coding skills necessary.
- Construction and export of hardware description (.xsa) from Vivado.
- Highlighting small details in Vivado that have a big impact on downstream PetaLinux and Vitis tools.
- Optimizing the bitstream for quicker boots and baremetal development

- Baremetal driver and application development on the ARM A5 using Vitis SDK.
- Using PetaLinux tools to construct embedded Linux Operating Systems
- Organizing root file system and network utilities
- Handling MAC addresses on networks that support multiple devCards
- Polling application to interact with custom hardware.
- Interrupt driven application to interact with custom hardware using various embedded Linux driver models.
- Measuring minimum, maximum, mean, and variance of interrupt latencies using various driver models to guide the embedded system architect..
- Discussion on development trade-offs (complexity versus performance versus time to market)
- Creating a System Project with Linux and baremetal applications.
- Safe and speedy remote updates using custom scripts.
- Performance of non-volatile storage via QSPI, SATA, and SD-CARDS, compared against a volatile root file system as a RAM disk.
- Mounting non-volatile storage.
- Getting Vital Product Data (VPD) out of I2C (e.g., ethernet MAC address)
- Ancillary skills using Linux command line utilities and tools to aid in rapid development.
- This training will save a typical project 2 months of time.

Register Today

Morgan Advanced Programmable Systems, Inc., delivers public and private courses in locations throughout the central US region; including Iowa, Illinois, Kansas, Minnesota, Missouri, Nebraska, North Dakota, South Dakota and Wisconsin.

Visit **morgan-aps.com/training**, for full course schedule and training information.

Morgan Advanced Programmable Systems

You must have your payment information available when you enroll. We accept credit cards (Visa, MasterCard, or American Express) as well as purchase orders and Xilinx training credits.

Student Cancellation Policy

- Students cancellations received more than 7 days before the first day of class are entitled to a 100% refund. Refunds will be processed within 14 days.
- Student cancellations received less than 7 days before the first day of class are entitled to a 100% credit toward a future class.
- Student cancellations must be sent here.

Morgan A.P.S. Course Cancellation Policy

- We regret from time-to-time classes will need to be rescheduled or cancelled.
- In the event of cancellation, live on-line training may be offered as a substitute.
- Morgan A.P.S. may cancel a class up to 7 days before the scheduled start date of the class; all students will be entitled to a 100% refund.
- Under no circumstances is Morgan A.P.S. responsible or liable for travel, lodging or other incidental costs. Please be aware of this cancellation policy when making your arrangements.
- For additional information or to schedule a private class contact us here.

Online training with real hardware

During the Covid-19 period, some companies do not allow their staff to participate in live in- person training.

- Consequently, Morgan Advanced Programmable Systems, Inc. has set up a training VPN where engineer participants can take classes online using the same computers and devCards used during in-person training.
- Even better, and upon request, you can use these computers after hours on training days to experiment with labs. This is not possible for in-person training.
- Additionally, just like in-person training, the laptops and devCards, tools, OS, and licensing are setup in advance.
- In some ways, live online-training is better than in-person…for example, you can grant the instructor permission to look at your Vivado, PetaLinux terminal, or Vitis for extended periods of time if your lab is not going exactly has planned to a missed step.
- This is often more comfortable than two engineers crowding around a laptop screen.
- Taking remote training also allows you to learn some tips and tricks for working remote. Whether your devCard is in the lab down the hall, or across the world via VPN, you can control your Xilinx based device quickly and efficiently, while learning to remotely troubleshoot problems.